

# Scheduling algorithms for automatic control systems for technological processes

A S Chernigovskiy, R Yu Tsarev, D V Kapulin

Siberian Federal University, 79, Svobodny ave., Krasnoyarsk, 660041, Russia

E-mail: achernigovskiy@sfu-kras.ru

**Abstract.** Wide use of automatic process control systems and the usage of high-performance systems containing a number of computers (processors) give opportunities for creation of high-quality and fast production that increases competitiveness of an enterprise. Exact and fast calculations, control computation, and processing of the big data arrays – all of this requires the high level of productivity and, at the same time, minimum time of data handling and result receiving. In order to reach the best time, it is necessary not only to use computing resources optimally, but also to design and develop the software so that time gain will be maximal. For this purpose task (jobs or operations), scheduling techniques for the multi-machine/multiprocessor systems are applied. Some of basic task scheduling methods for the multi-machine process control systems are considered in this paper, their advantages and disadvantages come to light, and also some usage considerations, in case of the software for automatic process control systems developing, are made.

## 1. Introduction

Nowadays an enterprise cannot exist in the competitive environment and even function at all if processes and operations control of production are not carried out in an automatic mode by control systems [9]. An automatic process control system (APCS) is an integral part of production at the large modern enterprises. Application of APCS not only allows reducing usage of human hand labor, but also promotes improving production quality [2, 6]. As a result, requirement and necessity for use of APCS are obvious. Also, it is not necessarily to use APCS in production everywhere; it is possible to carry out automation step by step, starting with the labor-consuming processes, and processes requiring high accuracy.

With high rates of technical progress in the field of computing, microprocessors become more and more powerful, and their cost becomes less. In this regard, the modern APCS the multiprocessor and multicomputer systems begin to be used more often. They allow increasing reliability of APCS on the whole, because in case of a failure of one unit, the system will not cease to function because of components redundancy [14].

Redundancy is especially necessary in case of real-time operation, as the organization of concurrent and multithreaded processes is possible because of it [1]. But not only correctly organized architecture of hardware is necessary for achievement of the maximum productivity. As well the software developed for APCS, using multi-machine and multiprocessor computing systems, has a great influence on the speed of receiving results and correct controlling actions, on the speed and quality of input data processing [7]. For this purpose it is necessary to select correctly the concurrent processes proceeding in APCS and to assign to their available computing devices of system so that to receive minimum possible time of result receiving [11]. For this purpose, it is possible to use task scheduling algorithms for the distributed computer systems. These algorithms are used to create such task schedule or operations execution which in case of execution on APCS gives minimum time of result receiving [16]. In this paper, basic task scheduling algorithms, which can be used in case of design of automatic process control systems, are considered.

The algorithms given below are intended for the one problem solution: making of such plan in which execution time of all tasks will be minimum (it is denoted as  $C_{\max}$  in a scheduling problem definition). In

such case, considered algorithms execute it in different methods and using different models. Scheduling is carried for identical machines, which are machines that have identical productivity. We will carry out the review of some algorithms and will show their applicability boundaries, as well as advantages and disadvantages.

## 2. Scheduling algorithms' inputs and outcomes

In this section we consider well-known scheduling algorithms that have been proven to solve effectively different practical scheduling problem and can be applied for automatic control systems for technological processes.

### 2.1. Hu's algorithm

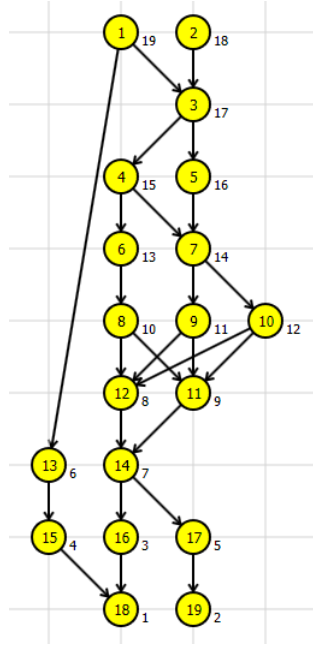
First of all, it is necessary to consider that many algorithms use the precedence constraints among tasks to show dependence of one task on others and impossibility of their execution, the prior will not be executed yet. Usage of the oriented graphs is represented as the most convenient method. Hu's algorithm is not an exception here; it is applicable for the acyclic oriented graphs in which each vertex has only one immediate successor, except finite vertex (intree) [8]. It is possible to call such graph as a «reversed» tree, its difference is in that a tree graph has one vertex without the entering arcs and some leaving arcs (root); in this model, the reverse case is used - there is one vertex without the leaving arcs (sink) with a set of the entering [3]. Also restriction on the processing time of each task is introduced, they need to have equivalent duration in one or in  $p$  of time units ( $p_j = p$  or  $p_j = 1$ ). Thus, it can be written a scheduling problem definition for Hu's algorithm as follows,  $P|intree, p_j = p|C_{max}$  (or  $P|intree, p_j = 1|C_{max}$ ) [8] that means: making of such task execution plan that concerns precedence, processing times are identical, that gives minimum completion time of all tasks set for finite number of identical machines (processors). It is proved that the plan produced by Hu's algorithm is optimal for the above described problem definition. Time complexity of algorithm is  $O(n)$ .

It is very convenient to use this algorithm for such processes which have a large number of input data and afterwards are processed in parallel and then as result give only one value or the controlling action, because the structure of such process completely matches structure of a graph which each vertex has only one immediate successor. Also, it is very convenient to apply it in that case, when there is no opportunity to measure runtime of each separate task and it is easier to take each of them for a time unit if the schedule received such methods afterwards does not arrange with productivity results. It is possible to perform further optimization or to use other algorithms. Hu's algorithm is easy to use and does not require big computations; it can be applied directly under system design to provide satisfactory software modules placement result. Sometimes, during the design, developers while planning placement of the software modules use the methods explained in Hu's algorithm unconsciously. It is possible to tell that this algorithm is result of natural attempt of computation scheduling and system optimization for the multi-machine (multiprocessor) computing system.

As a disadvantage, of course, it is necessary to consider that this algorithm does not provide preemptive schedules (with interruptions), but in certain cases it is not necessary or even technically impossible. But, nevertheless, Hu's algorithm is considered one of the simplest and basic algorithms for task scheduling with the described «reversed» tree structure. One more disadvantage is equal duration restriction of all system tasks, it can lead to some system idle time, but as it is mentioned, sometimes it is simply impossible to avoid.

### 2.2. Coffman-Graham algorithm

The optimality of this algorithm is proved for  $P2|prec, p_j = p|C_{max}$ , ( $P2|prec, p_j = 1|C_{max}$ ) [4, 5, 15], all designations match with those designated above, the difference only in  $P2$ . Here  $P2$  is an optimality for two machines (processors). As it is possible to note, one immediate successor restriction (intree absence in a problem definition) for each graph vertex is lifted in this algorithm (Fig. 1), but essential restriction is introduced, which is optimal only for two machines. However, it is proved that it solves not only the minimum makespan problem, but also finds minimum value of the tasks (operations) completion moments problem ( $\sum C_j$ ), having solved both of these tasks  $C_{max}$  and  $\sum C_j$ , it is possible to receive the so-called *ideal* schedule. It means that this algorithm one of the best in case of the given restrictions. Time complexity  $O(n^2)$  is higher than Hu's algorithm have, because of two stages of scheduling: labeling each task and following tasks assignment for each machine according to received labels. Each task can use the first labeling stage for obtaining the task list for the following use in algorithms which are intended for task list scheduling which will be considered further.



**Figure 1.** The graph example acceptable for Coffman-Graham algorithm.

As it was already told, this algorithm is one of the best for two-machine systems, the schedule received with its help is considered *ideal*, and also, unlike Hu's algorithm, it can be applied to any of the acyclic directed graphs. *Ideal* means that the result schedule will have not only the minimum length, but also ending time of each task in the schedule will be as minimal as possible (Fig. 2). Also, as well as Hu's algorithm, it is useful while it is impossible to estimate task duration and it is possible to use unit time restriction for tasks.

The main delimiter is non-optimality for  $m \geq 3$  machines and not non-preemptiveness; some advantages of non-preemptive schedules were mentioned above.

	0	1	2	3	4	5	6	7	8	9	10
1	1	3	5	7	10	8	11	14	17	19	
2	2	13	4	6	9	15	12		16	18	

**Figure 2.** A schedule derived by means of Coffman-Graham algorithm from the graph in fig. 1.

### 2.3. LPT (Largest Processing Time) algorithm

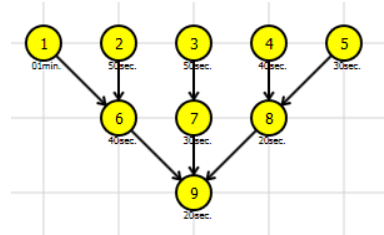
The LPT algorithm solves the  $P \parallel C_{\max}$  problem. From a problem definition, it is visible that the precedence relation is not considered, and some independent tasks list is processed. The task list can be created by Coffman-Graham's labeling algorithm, considering dependences, or any other possible way, while analyzing the structure of the graph vertexes precedence. The LPT algorithm is seldom applied in a pure form; more often it is used for combinatory optimization. With optimizations and using reasonable tasks choice, which scheduling gives the best result, this algorithm is capable to show good efficiency results. Time complexity is  $O(n \cdot \log(n))$ .

The LPT algorithm is very simple and evident in use that is an important advantage. Also, unlike the first two considered algorithms, it can be applied to any duration tasks, without  $p_j = p$  type restrictions, and it allows avoiding those idle times which can be received as a result of algorithms with identical task durations. The LPT algorithm plays an important role in further optimization. It is used in many modified algorithms. Therefore, it should not be treated as a non-optimal solution. It can be applied in the way that the system productivity will be increased on the whole.

An important restriction is the impossibility of the direct algorithm use for graph structures, but only for independent tasks lists which need to be obtained independently. Also, its relative optimality is equal to  $t_{\text{LPT}}(TS, m) / t_{\text{opt}}(TS, m) \leq 4/3 - 1 / (3m)$  [4, 5], where  $t_{\text{LPT}}(TS, m) = C_{\max}$  of the dchedule received by LPT algorithm for the independent tasks list  $TS$  for  $m$  of machines,  $t_{\text{opt}}(TS, m) = C_{\max}$  of the optimal schedule for the same  $TS$  and  $m$  values. It means that the non-optimal decision at worst equals  $4/3 - 1 / (3m)$ , where  $m$  number of machines (handlers) in certain cases can be received.

#### 2.4. McNaughton's algorithm

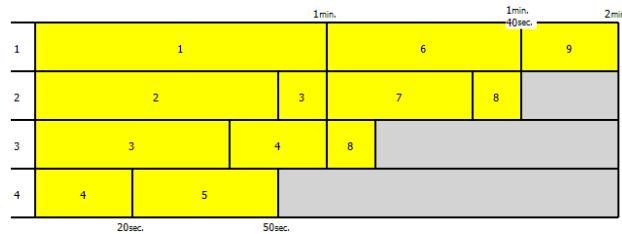
McNaughton's algorithm is optimal for the  $P|pmtn|C_{\max}$  problem [10]. For a start, it is worth mentioning such concept as preemptive scheduling. Preemptive scheduling considers the possibility to interrupt a task and to continue its performing later using another machine (processor). In this case, it is also necessary to consider the information transfer time subsequently, as well as some delays at interruptions, but nevertheless, scheduling algorithms consider ideal cases (Fig. 3). Here, *pmtn* means what it was told earlier, it indicates that it is possible to split (interrupt) tasks and then to carry out them using other machine. Of course, it should be taken into account, whether there is a possibility of tasks interruption in that industrial control system for which the software is projected, at technical or program impossibility of interruptions this algorithm won't manage to be applied. Time complexity is  $O(n^2)$ .



**Figure 3.** The graph example acceptable for McNaughton's algorithm.

The undoubted advantage of this algorithm is preemptiveness, it is the possibility accounting tasks division and the subsequent possible execution using another machine. Also, this algorithm can use non-unit task durations, therefore, the plan received with its help will not contain idle times, which can arise because of restrictions on task durations (Fig. 4). Therefore, in contrast to the LPT algorithm, this algorithm considers a possibility of tasks interruption and, also, allows avoiding the idle times caused by impossibility of task splitting into parts and, as a result, of emergence of an idle time of longer task processing.

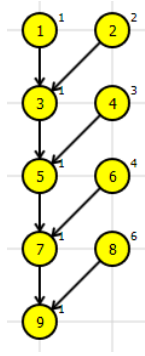
This method has the same disadvantages as LPT, i.e. it cannot be applied for a graph structure directly, although it is proved to be optimal for an independent tasks list. It is recommended to follow the same procedure to construct the task list of, as for the LPT algorithm.



**Figure 4.** A schedule derived by means of McNaughton's algorithm from the graph in fig. 3 for four processors.

#### 2.5. Muntz-Coffman algorithm

The algorithm is intended for  $P|pmtn, intree|C_{\max}$  and  $P2|pmtn, prec|C_{\max}$  problems solution [12, 13]. It means that it is optimal for a graph where each vertex has only one immediate successor (Fig. 5). Thus, it is possible to split task execution and then continue with another computer (processor). In addition, it is optimal for two machines for any acyclic kind of a graph.

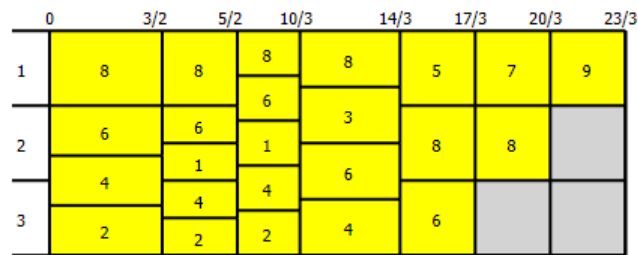


**Figure 5.** The graph example acceptable for Muntz-Coffman algorithm

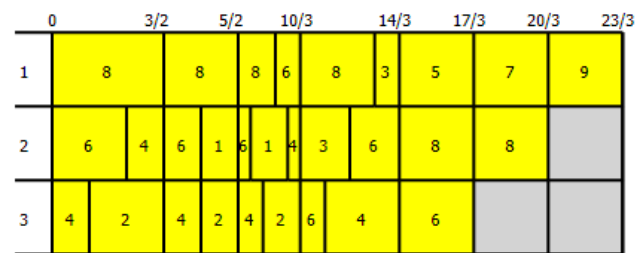
Moreover, it is possible to receive the execution task schedule with processor resource sharing a based on preemptive schedule, if such opportunity is available. In this case, it is not important what kind of duration vertexes have – number of time units or normal time – the algorithm can work with any of them.

This algorithm contains all advantages of the previous algorithms, it is optimal for intree (as well as Hu's algorithm), and also for any kind of the acyclic graph for two machines (as well as Coffman-Graham's algorithm). It operates with unit time tasks and with usual time durations (as well as LPT algorithm); it allows creating preemptive schedules with interruptions (as McNaughton's algorithm) (Fig. 6), as well as machine (processor) resource sharing schedules (Fig. 7).

The same delimiter, as in the first two algorithms have is impossibility of application to any kind of the directed graph structure. The first restriction – the graph needs to be a «reversed» tree type; the second – it can be any kind of graph, but the schedule will be optimal for a two-machine system.



**Figure 6.** A preemptive schedule derived by means of Muntz-Coffman algorithm from the graph in fig. 5.



**Figure 7.** A resource sharing schedule derived by means of Muntz-Coffman algorithm from the graph in fig. 5.

In Table 1, it is possible to find distinguishing features of observed scheduling algorithms. Table 1 covers these features to facilitate the selection of the proper algorithm to solve the scheduling problem.

**Table 1.** Distinguishing features of scheduling algorithms

	Hu's	Coffman-Graham	Muntz-Coffman	LPT	McNaughton
Nonpreemptive	+	+		+	
Preemptive			+		+

Unit processing times	+	+	+		
Any kind of processing time				+	+
Simple schedule structure	+	+		+	
Machine resources sharing scheduling			+		

---

### 3. Conclusion

In this paper, the main task or operations scheduling algorithms of the software of APCS for execution on the distributed multiprocessor or multi-machine system are considered. In the paper, the review of basic algorithms of scheduling is made, advantages and disadvantages of each of them are revealed; some recommendations about application of algorithms for design of the software of APCS are given. Boundaries of applicability of each of the algorithms, as well as time complexity of their execution are shown in the article.

Algorithms features considered in this article give sufficient information for software developing of APCS, which optimally uses computing resources, for the purpose of receiving a minimum result receiving time. However, eventually, a necessary scheduling algorithm choice, as well as the design need, is to be applied reasonably by experts and system designers. In the making of the optimum task execution plan, it is necessary to consider the structure of processed data, its transmission flows and many other features of APCS functions program implementation. The set of hardware and the software features metering is a ticket to success of the best APCS composition and the minimum time of receiving the final product.

### Acknowledgements

This work was supported by the Ministry of Education and Science of the Russian Federation in the framework of the Federal target program «Research and development of priority directions of development of the scientific-technological complex of Russia for 2014-2020» (agreement № 14.578.21.0116, unique ID project RFMEFI57815X0116).

### References

- [1] Al-Asaad H 2014 Real time scheduling of multiple executions of tasks to achieve fault tolerance in multiprocessor systems *AUTOTESTCON Proc. (St. Louis, United States, 15-18 September 2014)* 323–328
- [2] Cai Z, Hong R and Cui J 2015 Control system design for induction hardening machine based on PLC and HMI *Heat. Treat. Met.* **40** 224–227
- [3] Chernigovskiy A S, Tsarev R Y and Knyazkov A N 2015 Hu's algorithm application for task scheduling in N-version software for satellite communications control systems *Int. Siberian Conf. on Control and Communications, SIBCON 2015: Proc. (Omsk, Russia, 21-23 May 2015)* 1–4
- [4] Graham R L 1966 Bounds for certain multiprocessing anomalies *Bell Syst. Tech. J.* **45** 1563–1581
- [5] Graham R L 1969 Bounds on multiprocessing timing anomalies *SIAM J. Appl. Math.* **17** 263–269
- [6] Hamm C and Schaefer E 2015 Numerical control for ultra-precision machining *Proc. of the 15th Int. Conf. of the European Soc. for Precision Eng. and Nanotechnology, EUSPEN 2015 (Leuven, Belgium, 1-5 June 2015)* 3–7
- [7] Hashimoto K, Tsuchiya T and Kikuno T 2000 New approach to fault-tolerant scheduling using task duplication in multiprocessor systems *J. Syst. Software* **53** 159–171
- [8] Hu T C 1961 Parallel Sequencing and Assembly Line Problems *Oper. Res.* **9** 841–848
- [9] Grigorenko O, Stratan D and Sedykh J 2015 Current trends in the strategy of innovative development of industries in Russia *Mediterr. J. Soc. Sci.* **6** 364–370
- [10] McNaughton R 1959 Scheduling with deadlines and loss functions *Manag. Sci.* **6** 1–12
- [11] Mertziotis G B, Shalom M, Voloshin A, Wong P W H and Zaks S 2015 Optimizing busy time on parallel machines *Theor. Comput. Sci.* **562** 524–541
- [12] Muntz R R and Coffman E G Jr 1969 Optimal preemptive scheduling on two-processor systems *IEEE Trans. Comp.* **C-18** 1014–1020
- [13] Muntz R R and Coffman E G Jr 1970 Preemptive scheduling of real-time tasks on multiprocessor systems *J. ACM* **7** 324–338

- [14] Simevski A, Kraemer R and Krstic M 2014 Investigating core-level N-modular redundancy in multiprocessors *Proc. 2014 IEEE 8th Int. Symp. on Embedded Multicore/Manycore SoCs, MCSoc 2014 (Wakamatsu, Japan, 23-25 September 2014)* 175–180
- [15] Sinnen O 2014 Reducing the solution space of optimal task scheduling *Comput. Oper. Res.* **43** 201–214
- [16] Valckenaers P, Van Brussel H, Verstraete P, Saint Germain B and Hadeli 2007 Schedule execution in autonomic manufacturing execution systems *J. Manuf. Syst.* **26** 75–84